Sonatype

ASPECT SECURITY
Application Security Experts

Executive Brief: Addressing Security Concerns in Open Source Components

The Facts, The Issues, and Practical Solutions

Summary:

This executive brief summarizes the findings of an independent and comprehensive security review of the 31 most commonly used open source components and provides practical guidance and best practices for addressing security risks. Material was gathered from analysis conducted by researchers at Aspect Security[1] using data from *the Central Repository*, the industry's principal source for open source components, and a global Sonatype survey of 2,550 developers, architects, and industry experts.[2]

The analysis revealed widespread security vulnerabilities in components, creating risk for thousands of organizations using open source components in their software development efforts, including more than half of the Global 100. It also showed that few organizations take steps to mitigate risks posed by components that form the foundation of their most critical applications.

Interviews indicated a widely held view that popular open source components are of consistently high quality, given active community review to drive rapid defect resolution. This view overlooks fundamental ecosystem flaws, most notably the lack of centralized notification infrastructure alerting developers to the existence of flaws and the availability of new versions that correct them.

The authors are strong advocates of open source software and are active contributors to the open source community. To ensure maximum benefit from its use, organizations should take steps to minimize risks during software development, including improvements in awareness, policy, and enforcement. While our focus here is on open source software, many of the issues raised apply to all software.

A single vulnerable component can completely undermine the security of an application, expose valuable data assets, and jeopardize the integrity of an organization's software portfolio. The risks are very real, underscored by the growth and cost of cyber attacks. In 2011, successful cyber attack rates grew by 44% with an average time to resolution of 18 days and average cost of a data breach stands at $5.5 million.[3]

To ensure the integrity of the software development process, the authors recommend creating an inventory of open source components in custom applications, establishing controls throughout the development lifecycle, and monitoring the bill of materials of deployed applications for the discovery of new critical flaws.

Quantitative Analysis

Modern software relies heavily on open source:
More than 80 per cent of a typical software application is comprised of open source components and frameworks.

The Global 500 is at risk:
Collectively, Global 500 organizations downloaded more than 2.8 million insecure components in one year.

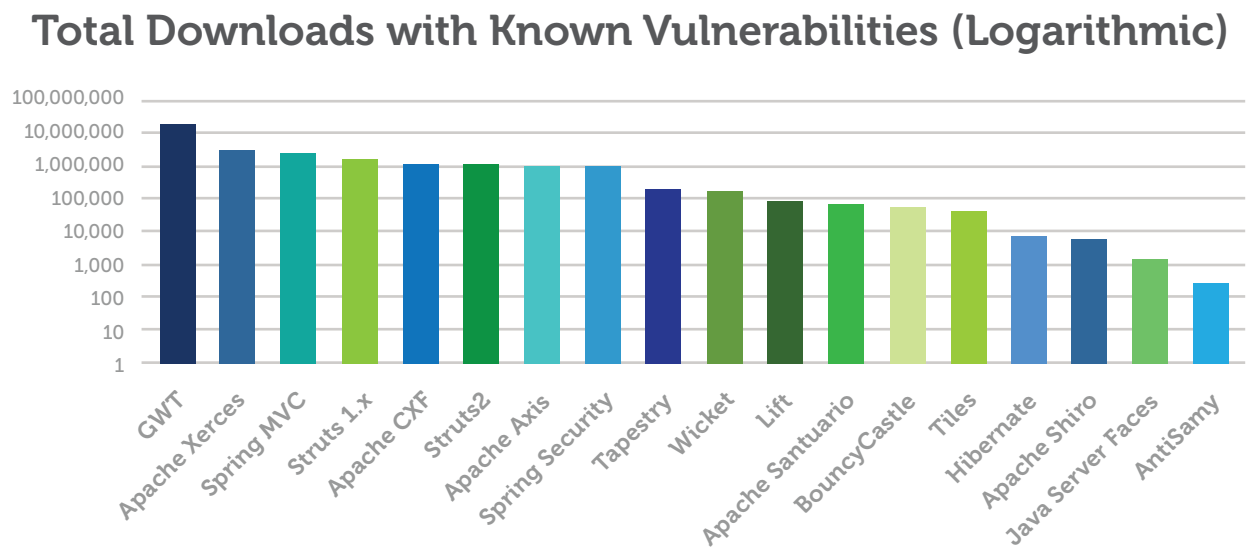
Financial services firms are the most exposed:
Global 100 financial services firms alone downloaded more than 567,000 insecure components in one year.

Many popular components have flaws:
There were more than 46 million downloads of insecure versions of the 31 most popular open source security libraries and web frameworks. Google Web Toolkit (GWT) was downloaded 17.7 million times with known vulnerabilities. Other popular vulnerable libraries downloaded included Xerces, Spring MVC, and Struts 1.x.

Users are not update aware:
One in three of the most popular components had older, vulnerable versions still being commonly downloaded, even when a newer version, with the security fix, was available.

Community scrutiny drives flaw discovery:
Open source security libraries are roughly 20 per cent more likely to have reported security vulnerabilities than other types of components. This is, at least in part, indicative of the effectiveness of broad community collaboration and active support.

The data shows that security vulnerabilities in open source libraries and frameworks are widespread. Yet most organizations do not appear to have a strong process in place for ensuring that the components they rely upon are up-to-date and free from known vulnerabilities.



Component Security Risks

Components (also referred to as binaries, artifacts, jars, or libraries) are software modules designed to perform commonly required functions including support for business logic, data access, resource management, communications, and user interface creation. Today's applications commonly use 30 or more components, which in turn might rely on dozens or hundreds of other components. As components run with the full privilege of the application, vulnerability in any given component can completely undermine the security of an entire application.

The impact of a component vulnerability often, but not always, depends on how the library is used by an application. Some libraries contain flaws that expose any application that leverages that library to compromise. A vulnerability in Struts 2 is an example of such a library.

In the last year, Struts 2 was downloaded over one million times by more than 18,000 organizations. In 2010, a class of weakness was discovered that allows attackers to execute arbitrary code on any Struts 2 web application. Following the initial discovery, Google and other researchers, have unearthed similarly critical flaws.

These Struts2 flaws do not require authentication or special skills to exploit. The following is an example exploit for the flaw documented in CVE-2010-1870.

http://example.org/struts2app/myaction?foo=%28%23context[%22xwork.MethodAccessor.denyMethodExecution%22]%3D+[...],%20%23_memberAccess[%22allowStaticMethodAccess%22]%3d+[...],%20@java.lang.Runtime@getRuntime%28%29.exec%28%27mkdir%20/tmp/PWND%27%29[...]27meh%27%29]=true

In this simple example, an exploit executes a 'mkdir' command (though, in reality an attacker could access any data, functionality, or privilege assigned to the application).

Vulnerabilities and Their Consumption is Common

Vulnerabilities are very common among the most popular components and frameworks.

This study revealed:

- 37% of the 1,261 versions of the 31 components studied contain a known CVE or OSVDB vulnerability.[4]
- 26% of downloads of these components were of versions with known CVE or OSVDB vulnerabilities.
- Popular component versions are only 10% less likely than less popular versions to contain a known vulnerability.
- Security libraries are roughly 20% more likely to have reported security vulnerabilities than web frameworks. This is likely due to increased security vigilance among these projects. It is probable that other components have vulnerabilities at similar (or higher) rates that are not yet known or reported.
- Often vulnerabilities exist in older versions of a component that have long since been fixed; however, vulnerable versions continue to be commonly used.

Why would someone use a vulnerable version when a newer, secure version is available? The answer is both revealing and troubling – the open source ecosystem both evolves rapidly and lacks centralized update notification infrastructure. Users are simply unaware that they are using flawed versions, and no efficient mechanism exists to alert them.

A recent global survey of 2,550 developers, architects, and managers revealed that organizations manage component updates on an ad-hoc basis, with no systematic mechanism for update awareness (see Figure 1).

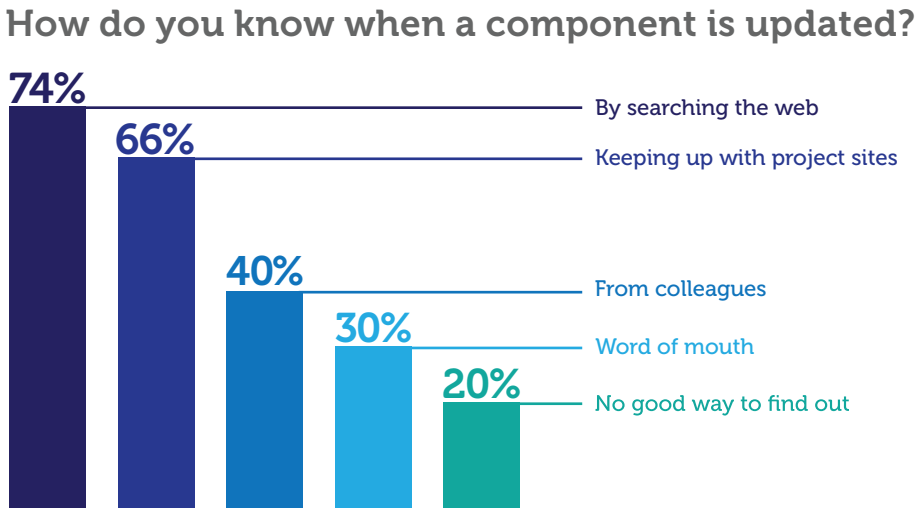


Figure 1: 2012 Sonatype survey of 2,550 developers, architects, and managers

Complex Dependencies Broaden Exposure, Increase Risk

The challenges facing software developers are compounded by the viral nature of the open source component ecosystem. A single component may be used in dozens or hundreds of others. A flaw in any given component version is effectively inherited by every component that depends on it. For instance, security vulnerability in Spring-beans 2.5.6 affected 1,447 dependent components. When Spring-beans was updated to fix the vulnerability, there was no process for updating the ecosystem, and hundreds of tainted components remain flawed.

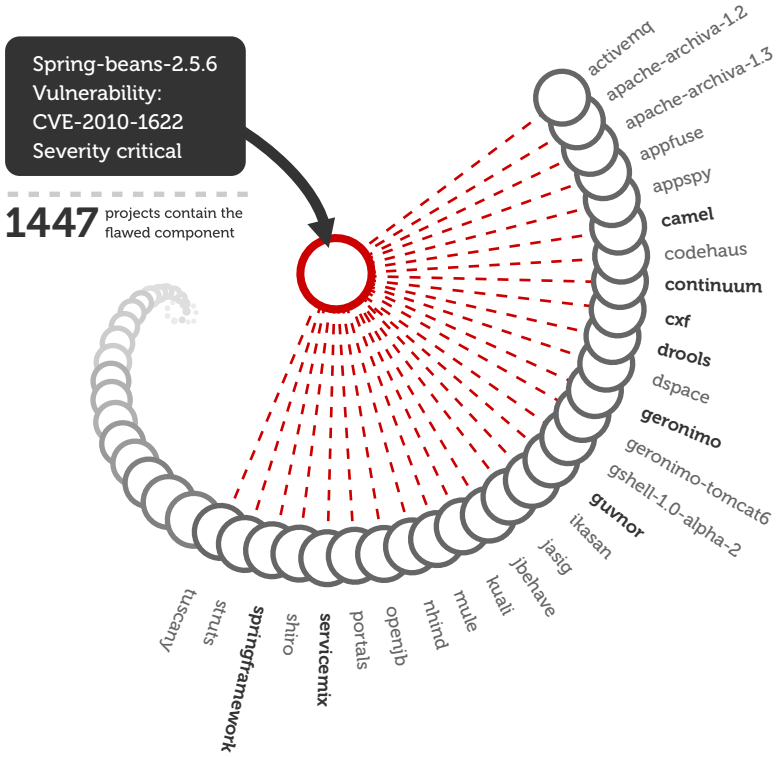


Figure 2: A vulnerability in Spring-beans 2.5.6 infected 1,447 other components and untold thousands of applications

Modern software applications are developed with unprecedented speed and are often incredibly powerful. They are also incredibly complex. Managing complex dependencies has the potential to yield security benefits. Effective dependency management processes would enable organizations to keep libraries more up-to-date, raise awareness of security vulnerabilities in libraries more quickly, and ensure that libraries are maintained in a healthy state.

The key to effective dependency management is dependency awareness. Survey data indicate that virtually all development organizations are handling updates to open source libraries on an ad-hoc basis. Compounding the problem, only 32 per cent of organizations maintain an inventory of the components and dependencies used in their production applications. When a new vulnerability is discovered, it is virtually impossible for these organizations to react in an appropriate manner to remediate the defect.

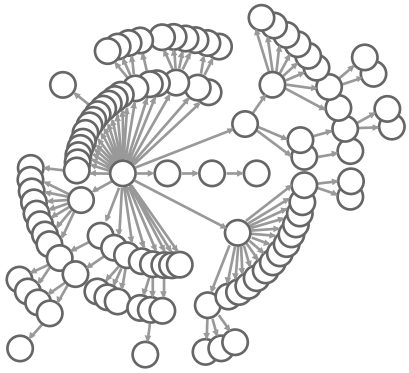


Figure 3: Even a simple software application can have dozens of multi-layered dependencies, which can obscure underlying issues.

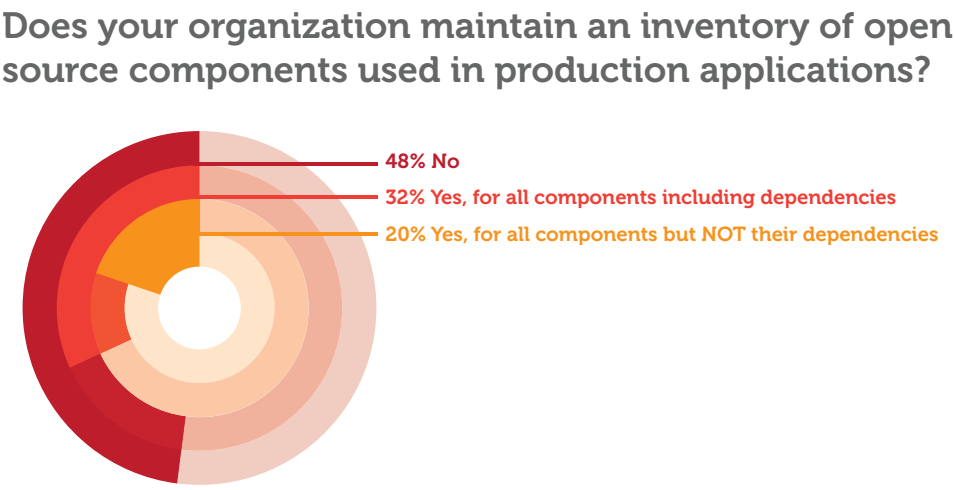


Figure 4: Only 32% of organizations maintain an inventory of the dependencies in their production applications, complicating issue resolution when a new vulnerability is discovered.

Recommendations

Given the prevalence of vulnerabilities in commonly used open source components, organizations must take steps to minimize risks during software development. This includes improvements in awareness, policy, and enforcement. To lessen the risks associated with using libraries and protect application portfolios, the following actions are recommended:

INVENTORY: Gather information about your current component usage

- Track component downloads and usage to understand consumption.
- Inventory internal component repositories to determine what is being distributed to development teams.
- Understand the software supply chain to determine what components and dependencies are being introduced to the organization.

ANALYZE: Understand vulnerabilities in applications and repositories

- Analyze key applications to uncover known security vulnerabilities.
- Analyze internal component repositories to discover vulnerable components.

CONTROL: Establish controls throughout the development lifecycle

- Establish policies regarding security, the use of viral licenses, and the out-of-date or out-of-version components.
- Eliminate or blacklist known vulnerable components in internal repositories.
- Establish mechanisms to prevent known flawed components from entering the organization.
- Implement controls in build and continuous integration systems to prevent inclusion of flawed components in software builds.

MONITOR: Maintain awareness of component updates

- Maintain an inventory of all components and dependencies used in production applications.
- Continuously monitor application bills of materials for updates and newly discovered vulnerabilities.

About Sonatype

Sonatype ensures the integrity of the modern software supply chain. Sonatype's tools and information services improve visibility and control over component-based software development, enabling collaboration while reducing the risks associated with security and licensing, and improving overall quality. Sonatype operates *the Central Repository*, the industry's primary source for open source components containing over 300,000 components, 200 million classes, and serving more than 4 billion requests per year from more than 60,000 organizations. Sonatype is a leader in open source projects, including Nexus, Apache Maven, m2eclipse and Hudson. The company was founded by Jason van Zyl, the creator of Apache Maven, and is privately held with investments from Accel Partners, Bay Partners, Hummer Winblad Venture Partners, and Morgenthaler Ventures. Visit www.sonatype.com or follow Sonatype on Twitter @SonatypeCM.

About Aspect Security

Founded in 2002, Aspect Security is a consulting firm focused exclusively on application security, ensuring that the software that drives business is protected against hackers. Aspect's engineers analyze, test and validate approximately 5,000,000 lines of critical application code every month. Aspect unearths more than 10,000 vulnerabilities every year across a wide range of technologies and architectures, and the company's practical recommendations dramatically improve clients' security posture. Aspect supports a worldwide clientele with critical applications in the government, defense, financial, healthcare, services and retail sectors. Aspect Security is a founding member of the Open Web Application Security Project (OWASP) and leads widely adopted projects such the OWASP Top Ten, WebGoat, the Application Security Verification Standard (ASVS), Risk Rating Methodology and Enterprise Security API (ESAPI). For more information, please visit www.aspectsecurity.com.

---

[1] Aspect Security , The Unfortunate Reality of Insecure Libraries https://www.aspectsecurity.com/news/press/the-unfortunate-reality-of-insecure-libraries/

[2] 2012 Sonatype Global Survey of 2,550 developers, architects, and executives. http://www.sonatype.com/people/2012/03/the-results-are-in-sonatype-2012-open-source-development-survey/

[3] Ponemon Institute: Second Annual Cost of Cyber Crime Study -- http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CEsQFjAA&url=http%3A%2F%2Fwww.arcsight.com%2Fcollateral%2Fwhitepapers%2F2011_Cost_of_Cyber_Crime_Study_August.pdf&ei=IKtjT5TGBYPX0QGJ15TjBw&usg=AFQjCNG6mLq6ly-i_3ZvfL8s3J_yQXu4XA and U.S. Cost of a Data Breach: http://www.symantec.com/about/news/release/article.jsp?prid=20120320_02&om_ext_cid=biz_socmed_twitter_facebook_marketwire_linkedin_2012Mar_worldwide__CODB_US

[4] CVE, is a dictionary of common vulnerabilities and exposures maintained by the Mitre Corporation -- (http://cve.mitre.org). OSVDB is the Open Source Vulnerability Database, an independent and open source database created and maintained by and for the community -- http://osvdb.org.